M16C/26

Using The M16C/26 Timer in PWM Mode

1.0 Abstract

PWM or Pulse Width Modulation is useful in DC motor control, actuator control, synthesized analog output, piezo transducers, etc. PWM produces a signal of (typically) fixed frequency and vary the width of the pulse to control a device or peripheral. The following article describes how to use the M16C/26 timer A in Pulse Width Modulation Mode.

2.0 Introduction

The Renesas M30262 is a 16-bit MCU based on the M16C/60 series CPU core. The MCU features include up to 64K bytes of Flash ROM, 2K bytes of RAM, and 4K bytes of Virtual EEPROM. The peripheral set includes 10-bit A/D, UARTS, Timers, DMA, and GPIO. The MCU has eight timers that consists of five Timer A's and three Timer B's. All five timer A's can operate as PWM's.

Timer A has the following additional modes of operation:

- Event Counter Mode
- Timer Mode
- One-Shot Mode

Figure 1 is a block diagram of timer A. The remainder of this document will focus on setting up timer A0 and A1 in PWM Mode.

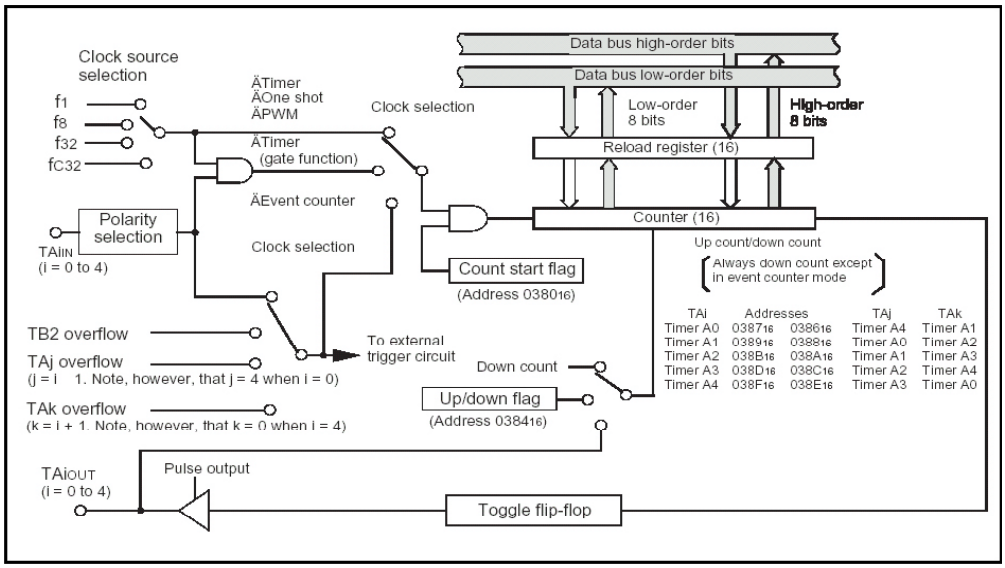


Figure 1 Bock Diagram of Timer A

3.0 PWM Mode Description

PWM Mode has two "sub" modes: 16-bit and 8-bit. In 16-bit mode, the value of the 16-bit Ai Timer register determines the pulse width and the frequency is fixed to fX_{in} / 65535. Therefore, the max frequency of the 16-bit PWM (assuming a 20MHz fX_{in}) is approx. 305 Hz. In 8-bit mode, the higher order 8-bits are used to determine the pulse width and the lower 8-bit the frequency, where the frequency is f_{in} / 255 or a maximum frequency of approx. 78,431 Hz. Note that the PWM output is free running and interrupts need not be serviced. In addition, the user has the option of triggering the start of the PWM output via the timer's TAiIN pin.

The pulse width (when the signal is at a high state) of the 16-bit PWM is:

Pulse width (high) = *n* / f_{in}, or % duty = *n* / 65535 * 100,

Where *n* is the value loaded into the Ai counter register.

The pulse width (when the signal is at a high state) of the 8-bit PWM is:

Pulse width (high) = *n* * (*m*+1) / f_{in}, or *n* / (255 * f_{PWM}), % duty = *n* / 255 * 100,

Where *n* is the value loaded into Ai counter register's high order address and *m* is loaded into Ai counter register's low order address.

The pulse width can be changed at any time by writing to the Ai counter register. However, during counting, the write only affects the reload register, and the counter register is updated on the next cycle.

Figure 2 and Figure 3 illustrate the timing for a 16-bit and 8-bit PWM.

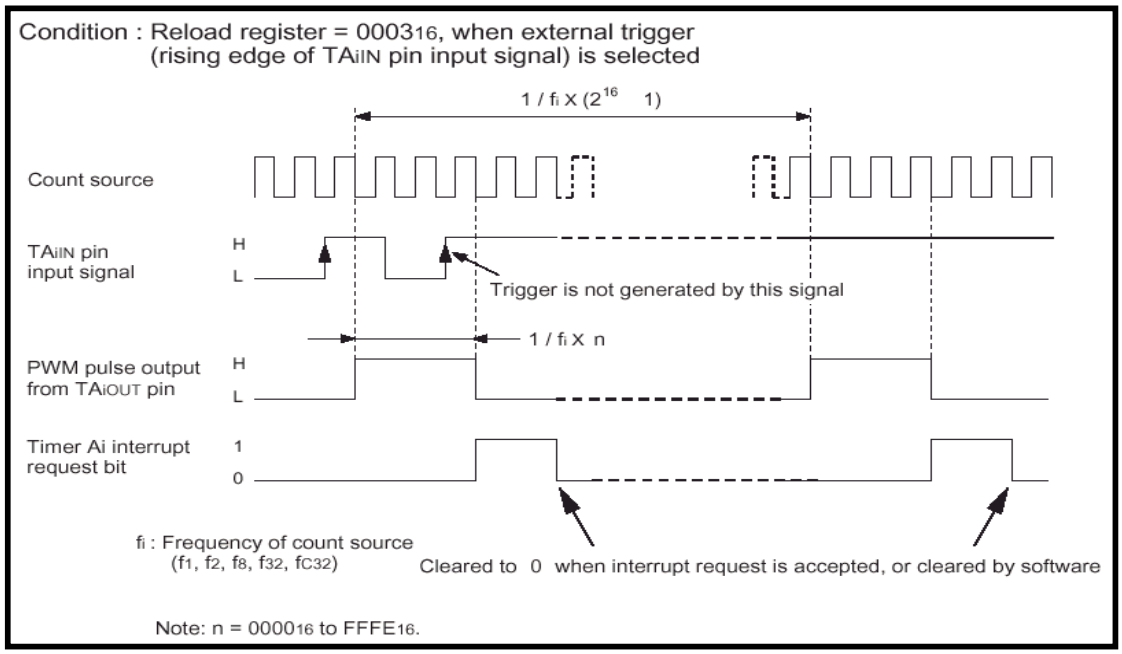


Figure 2 PWM Timing Example (16-bit mode)

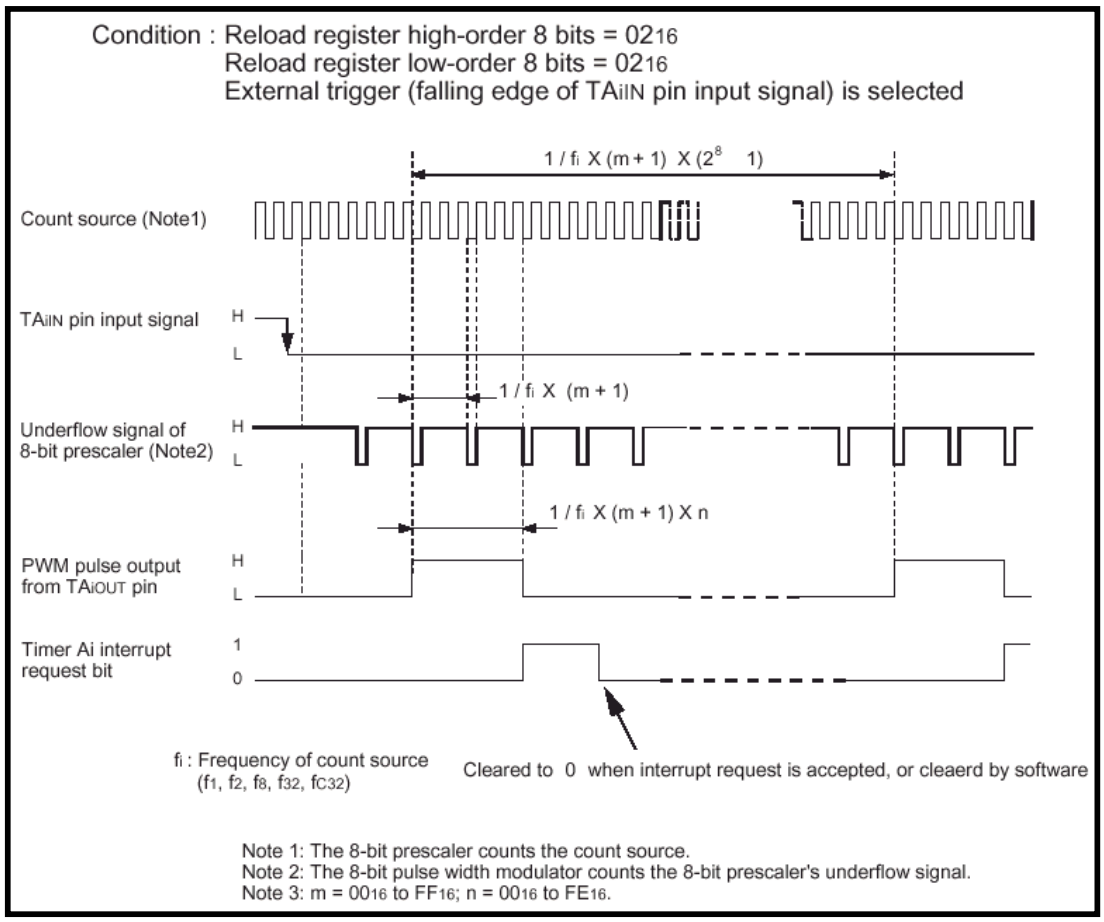


Figure 3 PWM Timing Example (8-bit mode)

4.0 Configuring PWM Mode

The steps to configure timer A for PWM mode are shown below.

1. Load Timer Ai register with pulse width value (16-bit) or frequency and pulse width value (8-bit).

Note: Note: For 16-bit 65535 is not valid for the pulse width value. For 8-bit 255 is not valid for the frequency and pulse width value.

2. Load the timer mode register, TAiMR
   - Select PWM mode: bits TMOD0 and TMOD1 = 1.
   - Set the MR0 bit = 1 for PWM Mode.
   - Clear the MR1 bit for a falling edge external trigger, or set it for rising edge.
   - Clear the MR2 bit to use the ‘count start flag’ as a trigger, or set it for external trigger.
   - Clear the MR3 bit for 16-bit PWM, set it for 8-bit PWM.
   - Select the clock source (f1, f/8, f/32, or fc/32): bits TCK0, TCK1 register.

3. Set the timer ‘interrupt priority level’, TAiIC (to zero if interrupts are not required)
4. Enable interrupts if required (CPU I flag set).
5. Set the ‘start count’ flag bit, TAiS in the ‘count start flag’ register, TABSR

For the most part, the order shown above is not important, but the count register should be loaded before the ‘start count’ flag is set. In addition, the priority level should not be modified when there is a chance of an interrupt occurring.

Figure 4 to Figure 7 show the registers for configuring Timer A for PWM.

Timer Ai mode register

b7	b6	b5	b4	b3	b2	b1	b0
					1	1	1

Symbol: TAiMR(i=0 to 4) — Address: 039616 to 039A16 — When reset: 0016

Bit symbol	Bit name	Function	R W
TMOD0	Operation mode select bit	b1 b0 1 1 : PWM mode	
TMOD1			
MR0	1 (Must always be 1 in PWM mode)		
MR1	External trigger select bit (Note 1)	0: Falling edge of TAiIN pin's input signal (Note 2) 1: Rising edge of TAiIN pin's input signal (Note 2)	
MR2	Trigger select bit	0: Count start flag is valid 1: Selected by event/trigger select bits	
MR3	16/8-bit PWM mode select bit	0: Functions as a 16-bit pulse width modulator 1: Functions as an 8-bit pulse width modulator	
TCK0	Count source select bit	b7 b6 0 0 : f1 0 1 : f8 1 0 : f32 1 1 : fC32	
TCK1			

Note 1: Valid only when the TAiIN pin is selected by the event/trigger select bit (addresses 038216 and 038316). If timer overflow is selected, this bit can be 1 or 0 .
Note 2: Set the corresponding port direction register to 0 .

Figure 4 Timer Ai mode register in Pulse Width Modulation Mode

Timer Ai register (Note 1)

Symbol	Address	When reset
TA0	038716,038616	Indeterminate
TA1	038916,038816	Indeterminate
TA2	038B16,038A16	Indeterminate
TA3	038D16,038C16	Indeterminate
TA4	038F16,038E16	Indeterminate

Function	Values that can be set	R	W
· Timer mode Counts an internal count source	000016 to FFFF16	○	○
· Event counter mode Counts pulses from an external source or timer overflow	000016 to FFFF16	○	○
· One-shot timer mode Counts a one shot width	000016 to FFFF16 (Note 2,6)	×	○
· Pulse width modulation mode (16-bit PWM) Functions as a 16-bit pulse width modulator	000016 to FFFE16 (Note 3,4,6)	×	○
· Pulse width modulation mode (8-bit PWM) Timer low-order address functions as an 8-bit prescaler and high-order address functions as an 8-bit pulse width modulator	0016 to FE16 (High-order address) 0016 to FF16 (Low-order address) (Note 3,5,6)	×	○

Note 1: Read and write data in 16-bit units.
Note 2: When the timer Ai register is set to 000016 , the counter does not operate and the timer Ai interrupt request is not generated. When the pulse is set to output, the pulse does not output from the TAiOUT pin.
Note 3: When the timer Ai register is set to 000016 , the pulse width modulator does not operate and the output level of the TAiOUT pin remains L level, therefore the timer Ai interrupt request is not generated. This also occurs in the 8-bit pulse width modulator mode when the significant 8 high-order bits in the timer Ai register are set to 0016 .
Note 4: When the set value =n, PWM period and "H" width of PWM pule are as follows:
PWM perild: $(2^{16} - 1)$/ fi, "H" width of PWM pulse: n / fi
Note 5: When the set value of upper-address=n and lower-address=m, PWM period and "H" width of PWM pule are as follows:
PWM perild: $(2^{8} - 1)x(m+1)$/ fi, "H" width of PWM pulse: (m+1)n / fi
Note 6: Use MOV instruction to write to this register.

Figure 5 Timer Ai register

One-shot start flag

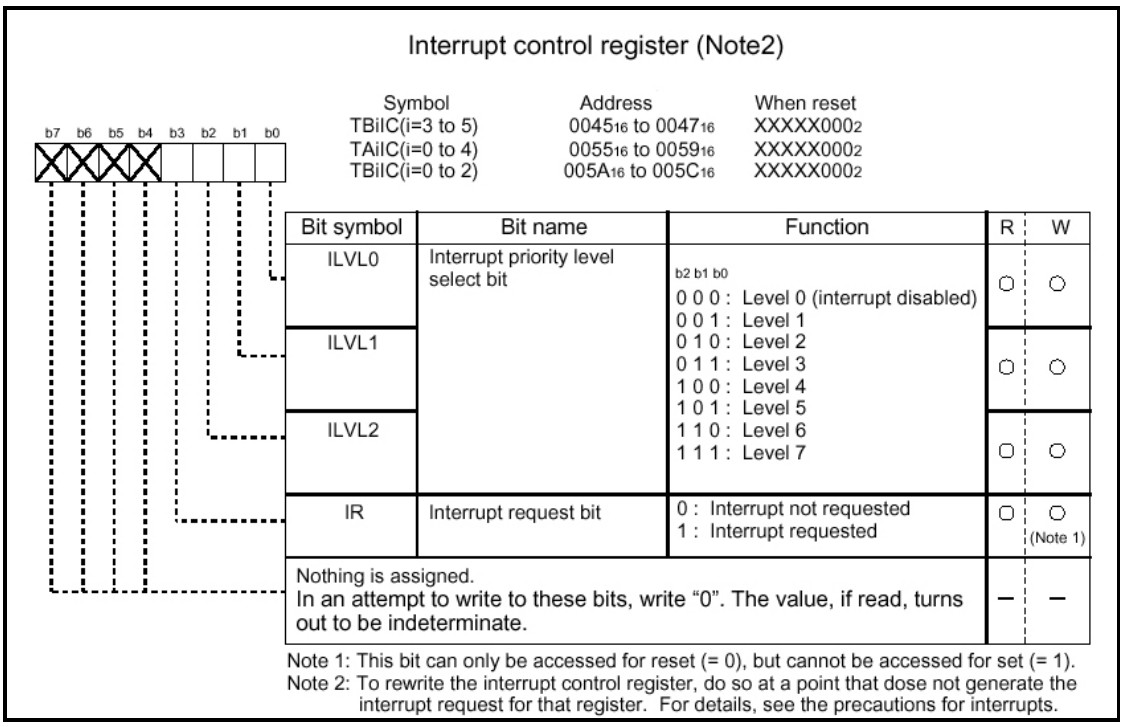
Symbol	Address	When reset
ONSF	038216	0016

b7	b6	b5	b4	b3	b2	b1	b0
		0					

Bit symbol	Bit name	Function	R	W
TA0OS	Timer A0 one-shot start flag	1 : Timer start When read, the value is 0	○	○
TA1OS	Timer A1 one-shot start flag		○	○
TA2OS	Timer A2 one-shot start flag		○	○
TA3OS	Timer A3 one-shot start flag		○	○
TA4OS	Timer A4 one-shot start flag		○	○
Reserved bit		Must always be set to 0	○	○
TA0TGL	Timer A0 event/trigger select bit	b7 b6 0 0 : Input on TA0IN is selected (Note) 0 1 : TB2 overflow is selected 1 0 : TA4 overflow is selected 1 1 : TA1 overflow is selected	○	○
TA0TGH				

Note: Set the corresponding port direction register to 0 .

Figure 6 Count start flag register



Interrupt control register (Note2)

Symbol	Address	When reset
TBiIC(i=3 to 5)	0045_{16} to 0047_{16}	XXXXX000$_2$
TAiIC(i=0 to 4)	0055_{16} to 0059_{16}	XXXXX000$_2$
TBiIC(i=0 to 2)	$005A_{16}$ to $005C_{16}$	XXXXX000$_2$

b7 b6 b5 b4 b3 b2 b1 b0

Bit symbol	Bit name	Function	R	W
ILVL0	Interrupt priority level select bit	b2 b1 b0 0 0 0 : Level 0 (interrupt disabled) 0 0 1 : Level 1 0 1 0 : Level 2 0 1 1 : Level 3 1 0 0 : Level 4 1 0 1 : Level 5 1 1 0 : Level 6 1 1 1 : Level 7	○	○
ILVL1			○	○
ILVL2			○	○
IR	Interrupt request bit	0 : Interrupt not requested 1 : Interrupt requested	○	○ (Note 1)
Nothing is assigned. In an attempt to write to these bits, write "0". The value, if read, turns out to be indeterminate.			–	–

Note 1: This bit can only be accessed for reset (= 0), but cannot be accessed for set (= 1).
Note 2: To rewrite the interrupt control register, do so at a point that dose not generate the interrupt request for that register. For details, see the precautions for interrupts.

Figure 7 Interrupt control register

5.0 Reference

Renesas Technology Corporation Semiconductor Home Page

http://www.renesas.com

E-mail Support

support_apl@renesas.com

Data Sheets

- M16C/26 datasheet, M30262eds.pdf

User's Manual

- KNC30 Users Manual, KNC30UE.PDF
- M16C/60 and M16C/20 C Language Programming Manual, 6020EC.PDF
- Application Note: Writing interrupt handlers in C for the M16C
- MSV30262-SKP or MSV-Mini26-SKP Quick start guide
- MSV30262-SKP or MSV-Mini26-SKP Users Manual
- MDECE30262 or MSV-Mini26-SKP Schematic

6.0 Software Code

A sample program written in C and compiled using the KNC30 compiler to illustrate how to set up a 16-bit PWM Mode on timer A0, and an 8-bit PWM on timer A1 is shown below. This program runs on the MSV30262 Starter Kit Board.

To get familiar with the PWM, try changing the clock source or switch to a different timer (i.e. TA2, TA3, etc).

```
/***********************************************************************
*
*       File Name: pwm.c
*
*       Content: Example program using Timer A in "PWM mode" .This program
*            is written for the 'Timer A PWM Mode' application note. Timer
*            A0 is set up as an 8-bit PWM (output on TA0out or P7_0),
*            timer A1 set up as a 16-bit PWM (output on TA1out or P7_2), and
*            the output varied. The frequency of the 16-bit PWM is set to 305Hz,
*            and the 8-bit PWM is set to 9765Hz.  The outputs can be viewed
*            on a scope. This program works with the MSV30262 starter kit board.
*            Note: Port P7_0 is an open drain output and may require a pull-up
*            resistor on target hardware.
*
*       Compiled with KNC30.
*
*       All timing based on 20 MHz Xtal
*
*      Copyright 2003 Renesas Technology America, Inc.
*      All Rights Reserved.
*=======================================================================
*       $Log:$
*=======================================================================*/
#include "sfr62.h"

#define  PWM8_CONFIG 0x67  /*  01100111 value to load into timer A0 mode register
                                |||||||_  TMOD0, TMOD1: PWM MODE SELECTED
                                ||||||____ MR0:          = 1 FOR PWM MODE
                                |||||_____ MR1, MR2:     EXTERNAL TRIGGER NOT SELECTED
                                |||________ MR3:          SET TO 1 FOR 8-BIT PWM
                                ||_________ TCK0, TCK1:   F8 SELECTED */

#define  PWM16_CONFIG 0x07  /* 00000111 value to load into timer A1 mode register
                                |||||||_  TMOD0, TMOD1: PWM MODE SELECTED
                                ||||||____ MR0:          = 1 FOR PWM MODE
                                |||||_____ MR1, MR2:     EXTERNAL TRIGGER NOT SELECTED
                                |||________ MR3:          SET TO 0 FOR 16BIT PWM
                                ||_________ TCK0, TCK1:   F1 SELECTED */

#define  CNTR_IPL 0x00   // TA0 AND TA1 interrupt priority level
int time_cnt;  // loop counter

//prototypes
 void init(void);

/***********************************************************************
```

```
Name:    main()
Parameters: none
Returns:  nothing
Description: initializes variables, then goes into an infinite loop. A
             simple delay loop is used to "slowly" increase the pulse widths
************************************************************************** */
void main (void)
{
  int pwm16;
  char pwm8;

  pwm16 = 100;  //16bit PWM changes slowly so give it a reasonable width to start
  pwm8 = 0;
  time_cnt = 0;
  init();
  while (1)
  {
        while(time_cnt <10000)
                time_cnt++;  // delay loop

      time_cnt = 0;
        pwm8++;                 // cannot read-modify-write on timer counter
        pwm16+=10;           // registers: while counting, value is indeterminate.
        if (pwm8 > 0xfe)     // the 8bit PWM value cannot exceed 0xFE
                pwm8 = 0;
        if (pwm16 > 0xfffe)  // the 16bit PWM value cannot exceed 0xFFFE
                pwm16 = 100;
        ta0h = pwm8;
        ta1 =pwm16;
  }
}
/************************************************************************
Name:  initial()
Parameters: none
Returns:  nothing
Description: Timer TA0 and TA1 setup for PWM mode.
************************************************************************ */
void init()
{
   ta0l = 0x00;    // PWM TA0 frequency set to 7,843 Hz

/* the following procedure for writing an Interrupt Priority Level follows that as described
in the M16C datasheets under 'Interrupts' */

// intialize TA0
   _asm ("   fclr i") ;      // turn off interrupts before modifying IPL
   ta0ic |= CNTR_IPL;      // use read-modify-write instruction to write IPL
   ta0mr = PWM8_CONFIG;
   _asm ("   fset i");
```

```
   ta0s = 1;  //start PWM

// intialize TA1
   _asm ("   fclr i") ;// turn off interrupts before modifying IPL
   ta1ic |= CNTR_IPL;          // use read-modify-write instruction to write IPL
   ta1mr = PWM16_CONFIG;
   _asm ("   fset i");

   ta1s = 1;  //start PWM

}
```

Keep safety first in your circuit designs!

- Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
  Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
- Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
- All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
  The information described here may contain technical inaccuracies or typographical errors.
  Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
  Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
- When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
- Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
- The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
  Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
- Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.